TEMA 1

INTRODUCCIÓN A LA PROGRAMACIÓN

1.1.- Concepto de ordenador y sistema operativo.

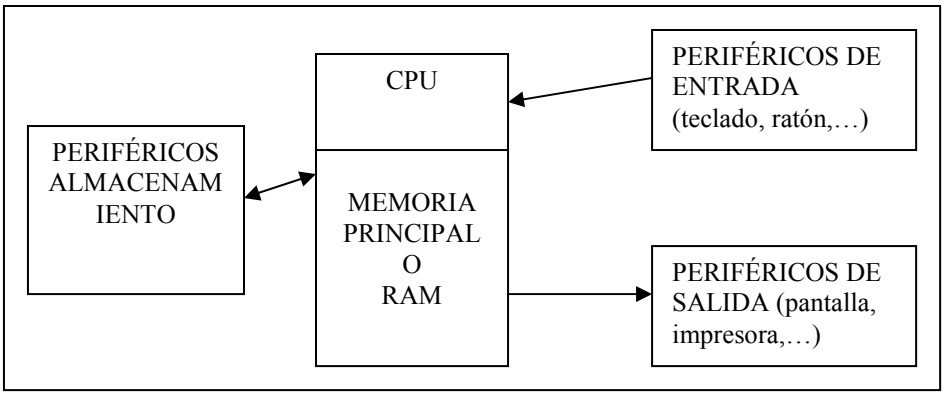
Un ordenador es una máquina creada por el hombre y por tanto no podrá realizar una tarea que no haya sido previamente determinada por él.

El ordenador no tiene inteligencia, todo lo que puede realizar son las siguientes operaciones básicas:

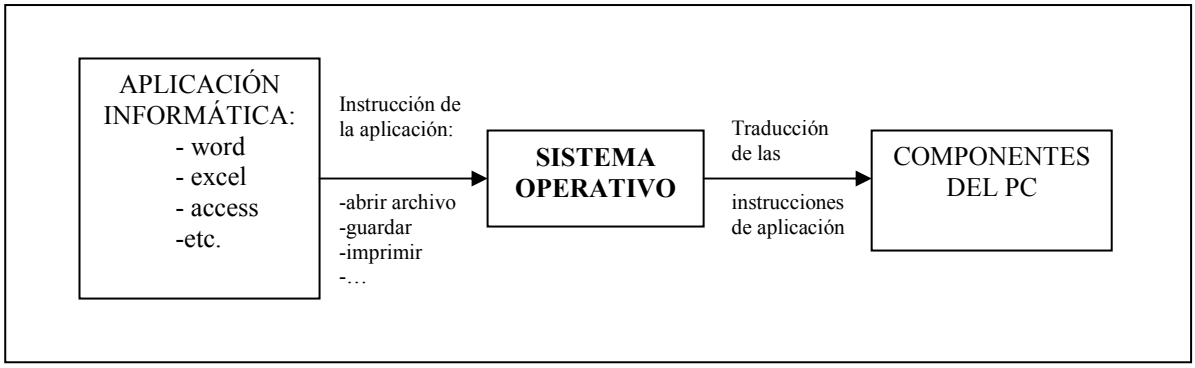
- Sumar y restar.
- Comparar dos valores (numéricos o alfanuméricos).
- Almacenar y recuperar información.

Todo ordenador se compone de HARDWARE y SOFTWARE.

- El HARDWARE sería: pantalla, teclado y unidad central o CPU con los componentes que incluye(memoria, micro,…), y demás periféricos.
- El SOFTWARE sería: el SISTEMA OPERATIVO de la máquina y los demás programas que estén instalados en el ordenador.



- Para que un ordenador funcione, es decir, realice alguna tarea, se necesita decirle lo que tiene que hacer y eso se logra mediante las **APLICACIONES INFORMÁTICAS** (parte lógica o software).
- El **SISTEMA OPERATIVO** es un software IMPRESCINDIBLE en el equipo. Es el intermediario entre la máquina y el usuario. También es el traductor entre las aplicaciones que usemos y la máquina, así como el gestor de los recursos de la máquina.



1.2.- Sistemas de procesamiento de la información.

- **INFORMÁTICA:** Ciencia que estudia el tratamiento automático de la información.
- **ORDENADOR**: Máquina capaz de recibir unos datos (INFORMACIÓN DE ENTRADA), procesarlos y mostrar unos resultados (INFORMACIÓN DE SALIDA).

- **PROGRAMA:** Para que un ordenador funcione, es decir, sea capaz de hacer algo con la información que le damos, deben dársele ordenes:

  "*coge el primer número que te doy,*
  súmalo con el segundo que te doy,
  y muéstrame esa suma en pantalla".

  Esas órdenes, el ordenador no las entiende (no entiende el lenguaje humano). Debemos dárselas en un lenguaje que él entienda: UN LENGUAJE DE PROGRAMACIÓN. Esas órdenes, en ese lenguaje es lo que se denomina un PROGRAMA. Por tanto, podemos decir también:

  PROGRAMA = Conjunto finito de instrucciones en un lenguaje de programación que definen un algoritmo.

  ALGORITMO = Conjunto finito de acciones ordenadas que se deben hacer para resolver un problema o realizar una tarea.

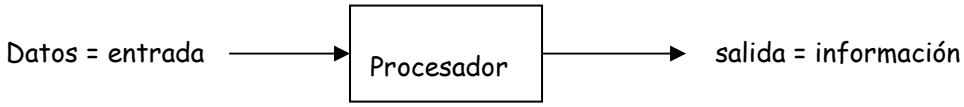
En el uso diario, datos e información son esencialmente sinónimos. Sin embargo los informáticos suelen hacer una diferencia:

- **Datos** se refieren a la representación de algún hecho, concepto o entidad real.
- **Información** implica datos procesados y organizados.

Un **sistema** en general se define como un conjunto de componentes conectados e interactivos, que tienen un propósito y una unidad total.

Sistema de procesamiento de información es un sistema que transforma datos brutos en información organizada y útil.

Los tres componentes básicos de un sistema de procesamiento de la información son: entrada, procesador, salida.



Es decir el procesador puede aceptar datos llamados entrada, procesarlos y producir una información denominada salida.

1.2.1.- La información y su representación.

- **INFORMACIÓN**: es un concepto abstracto, gracias al cual, las personas representan acontecimientos y hechos. Para ello, el hombre intenta representar gráficamente la información mediante distintos sistemas de representación como los SISTEMAS DE NUMERACIÓN existentes:
  - **Decimal**
  - **Binario**
  - **Octal**
  - **Hexadecimal**
  - …

1.2.2.- Sistemas de numeración.

Un sistema de numeración es el conjunto de símbolos utilizados para representar cantidades así como las reglas que rigen dicha representación.

Un número se puede representar por un conjunto ordenado de símbolos diferentes para cada uno de los sistemas existentes.

Los sistemas de numeración actuales se definen como posiciónales, es decir que el valor que representa cada símbolo depende de su valor absoluto y de la posición que ocupa dicha cifra con respecto a la coma decimal.

La base nos permite diferenciar el sistema de numeración con el que estamos trabajando. Dicha base es el número de símbolos utilizados para representar las cantidades, sin tener en cuenta el punto decimal.

Existe un teorema que nos da la pauta para comprender el sistema de numeración, es el Teorema Fundamental de la Numeración, que dice: "El valor decimal de una cantidad expresada en otro sistema de numeración viene dado por la fórmula":

$$..... + X_4 * B^4 + X_3 * B^3 + X_2 * B^2 + X_1 * B^1 + X_0 * B^0 + X_{-1} * B^{-1} + X_{-2} * B^{-2} + X_{-3} * B^{-3} + ...$$

Siendo X_i cada una de las cifras que componen el número , y B la base.

Los sistemas de numeración más utilizados son:

- **Sistema decimal:** Utiliza 10 símbolos (0, 1, 2, ... , 9) para representar todos los posibles valores. El nombre usado para cada uno de estos símbolos es "dígito". Cuando combinamos varios dígitos, tenemos un número. Su valor depende no solo del valor de cada uno de ellos sino de la posición que tiene dentro del conjunto.
- **Sistema binario:** Es el sistema de numeración que utiliza internamente el hardware del ordenador. Es un sistema que utiliza 2 símbolos (0, 1), es decir su base es 2. Cada uno de estos símbolos se denomina BIT. (Es la unidad mínima de información).
  La asociación de un número determinado de bits de uso muy común en el mundo informático es el siguiente:

1 BIT	un dígito 0 , 1
1 BYTE	8 BITS
1 KILOBYTE (KB)	1.024 BYTES = 2^{10} BYTES
1 MEGABYTE (MB)	1.024 KILOBYTES = 2^{10} KILOBYTES
1 GIGABYTE (GB)	1.024 MEGABYTES = 2^{10} MEGABYTE

- **Sistema octal:** Es un sistema de numeración que utiliza 8 dígitos (0, 1, ...,7) es decir que su base de numeración es 8. La conversión a binario es la siguiente:

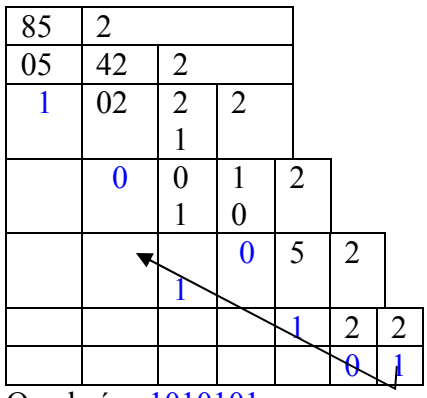
Binario	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

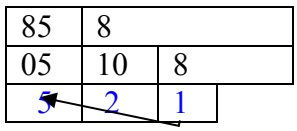
- **Sistema hexadecimal:** Es un sistema de numeración que utiliza 16 dígitos (0, 1, … 9, A, B, C, D, E, F) es decir su base de numeración es 16.

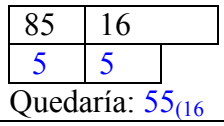
  Un inconveniente que representa el sistema binario es la utilización de gran cantidad de bits. Para solucionar este inconveniente se utiliza el sistema Hexadecimal, que permite trabajar cómodamente con el sistema binario, puesto que cada cifra equivale a 4 dígitos.

Hexadecimal	Binario	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

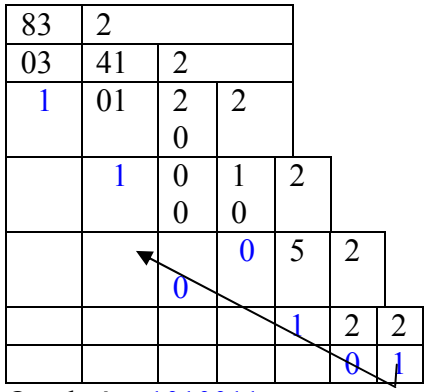
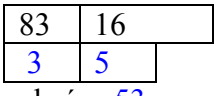
¿CÓMO CONVERTIR DE UN SISTEMA DE NUMERACIÓN A OTRO?

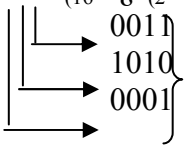
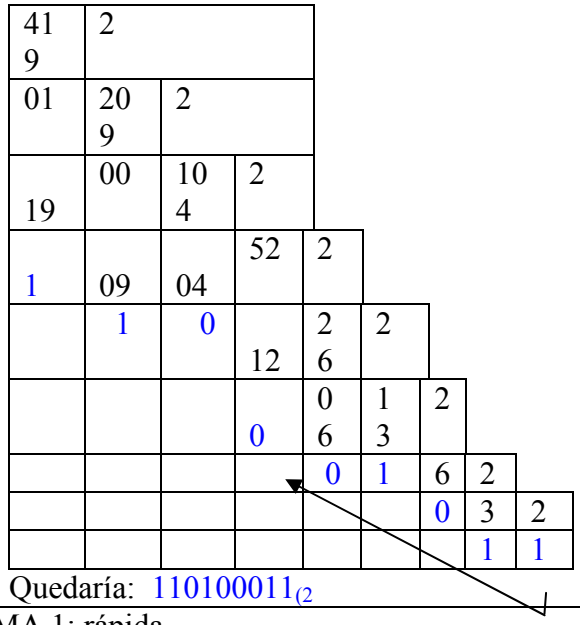
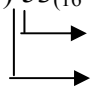
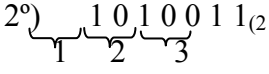
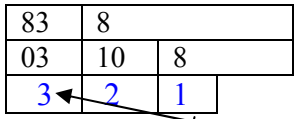
<table>
<tr><th>SISTEMA ORIGEN</th><th>SISTEMA DESTINO</th><th>PROCEDIMENTO</th></tr>
<tr><td>DECIMAL</td><td>Cualquier sistema</td><td>Se divide el nº entre la base del sistema destino (si es binario entre 2, entre 8 si es octal y entre 16 si es hexadecimal) y se siguen dividiendo los cocientes obtenidos hasta que el último cociente sea < que la base (es decir, hasta que ya no se pueda dividir más). Cuando terminemos de dividir se coge el cociente y todos los restos en el orden de abajo hacia arriba y ese es el nº en la base deseada.
Ej 1: $85_{(10} = ¿?\ _{(2}$
 
Quedaría: $1010101_{(2}$

Ej 2: $85_{(10} = ¿?_{(8}$
 
Quedaría: $125_{(8}$

Ej 3: $85_{(10} = ¿?_{(16}$
 
Quedaría: $55_{(16}$</td></tr>
<tr><td>CUALQUIER SISTEMA</td><td>DECIMAL</td><td>Se aplica el Teorema Fundamental de la Numeración.
Ej 1: BINARIO a DECIMAL:
$1010101_{(2} = 1x\ 2^6 + 0\ x\ 2^5 + 1x\ 2^4 + 0x\ 2^3 + 1x2^2 + 0x\ 2^1 + 1x\ 2^0 = 85_{(10}$</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Ej 2: OCTAL a DECIMAL:
$123_{(8} = 1x\ 8^2 + 2x\ 8^1 + 3x\ 8^0 = 64 + 16 + 3 = 83_{(10}$
Ej 3: HEXADECIMAL a DECIMAL:
$C1A_{(16} = Cx16^2 + 1x\ 16^1 + Ax16^0 = 12 * 256 + 16 + 10 *1=$
$= 3072 +26=3098_{(10}$</td></tr>
<tr><td>BINARIO</td><td>OCTAL</td><td>FORMA 1:
Se hacen grupos de 3 bits empezando por la derecha. Si al final el nº de bits que queda es inferior a 3, se rellena con 0 por la izda. Luego cada grupo se traduce al decimal correspondiente.
Ej: 0 0 1 0 1 0 1 0 1
1 2 5 ==> $125_{(8}$
FORMA 2:
1º) Pasar de BINARIO a DECIMAL:
Ej: $1010101_{(2}$ = aplicando teorema=$85_{(10}$
2º) Pasar de decimal a octal dividiendo entre 8:
<table><tr><td>85</td><td>8</td><td></td></tr><tr><td>05</td><td>10</td><td>8</td></tr><tr><td>5</td><td>2</td><td>1</td></tr></table>Quedaría: $125_{(8}$</td></tr>
<tr><td>BINARIO</td><td>HEXADECIMAL</td><td>FORMA 1:
Se hacen grupos de 4 bits empezando por la derecha. Si al final el nº de bits que queda es inferior a 4, se rellena con 0 por la izda. Luego cada grupo se traduce al decimal correspondiente.
Ej: 0 1 0 1 0 1 0 1
5 5 ==> $55_{(16}$
FORMA 2:
1º) Pasar de BINARIO a HEXADECIMAL:
Ej: $1010101_{(2}$ = aplicando teorema=$85_{(10}$
2º) Pasar de decimal a hexadecimal dividiendo entre 16:
<table><tr><td>85</td><td>16</td></tr><tr><td>5</td><td>5</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>Quedaría: $55_{(16}$</td></tr>
<tr><td>OCTAL</td><td>BINARIO</td><td>FORMA 1:
Se codifica cada dígito octal en binario con 3 bits y se colocan en el mismo orden en que estaban. Ese será el resultado.
Ej: $123_{(8} =¿?_{(2}$
011
010 } 001 010 011$_{(2}$
001
FORMA 2:
1º) Pasar de OCTAL a DECIMAL usando el teorema:
Ej: $123_{(8}$ = aplicando teorema=$83_{(10}$
2º) Pasar de decimal a binario dividiendo entre 2:
 
Quedaría: $1010011_{(2}$</td></tr>
<tr><td>OCTAL</td><td>HEXADECIMAL</td><td>FORMA 1: rápida
1º) Se pasa de octal a binario
2º) Se pasa de binario a hexadecimal
Ej: $123_{(8} =¿?_{(16}$
1º) $123_{(8}$
011
010 } 001 010 011$_{(2}$
001
2º) $\underbrace{1\,0\,1}_{5}\,\underbrace{0\,0\,1}_{3}\,1_{(2}$ ==> $53_{(16}$
FORMA 2:
1º) Pasar de OCTAL a DECIMAL usando el teorema:
Ej: $123_{(8}$ = aplicando teorema=$83_{(10}$
2º) Pasar de decimal a hexadecimal dividiendo entre 16:
 
Quedaría: $53_{(16}$</td></tr>
</table>

<table>
<tr>
<td>HEXADECIMAL</td>
<td>BINARIO</td>
<td>FORMA 1:
Se codifica cada dígito octal en binario con 4 bits y se colocan en el mismo orden en que estaban. Ese será el resultado.
Ej: $1A3_{(16}$ =¿?$_{(2}$
 
$0001\ 1010\ 0011_{(2}$
FORMA 2:
1º) Pasar de HEXADECIMAL a DECIMAL usando el teorema:
Ej: $1A3_{(16}$ = aplicando teorema=$419_{(10}$
2º) Pasar de decimal a binario dividiendo entre 2:
 
Quedaría: $110100011_{(2}$</td>
</tr>
<tr>
<td>HEXADECIMAL</td>
<td>OCTAL</td>
<td>FORMA 1: rápida
1º) Se pasa de hexadecimal a binario
2º) Se pasa de binario a octal
Ej: $53_{(16}$ =¿?$_{(8}$
1º) $53_{(16}$
 
$0101\ 0011_{(2}$
 
==> $123_{(8}$
FORMA 2:
1º) Pasar de HEXADECIMAL a DECIMAL usando el teorema:
Ej: $53_{(16}$ = aplicando teorema=$83_{(10}$
2º) Pasar de decimal a octal dividiendo entre 8:
 
Quedaría: $123_{(8}$</td>
</tr>
</table>

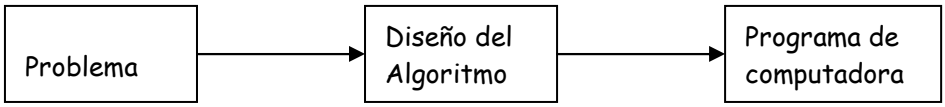
1.3.- Concepto de algoritmo.

El conjunto de instrucciones que especifican la secuencia de operaciones a realizar, en orden, para resolver un sistema especifico o clase de problema se denomina algoritmo.

"Algoritmo es la formula para la resolución de un problema".

Para realizar un proceso se debe suministrar al procesador un algoritmo adecuado.(Al cocinero una receta, al músico una partitura, ..etc.). Cuando el procesador es una computadora, el algoritmo ha de expresarse de una forma determinada que recibe el nombre de programa. Un programa se escribe en un lenguaje de programación y a la actividad de expresar un algoritmo en forma de programa se le denomina programación.

Cada paso en el algoritmo está expresado por medio de una instrucción en el programa. Por consiguiente, un programa consta de una secuencia de instrucciones, cada una de las cuales especifica las operaciones que debe realizar al computadora.



Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta.

En la ciencia de la computación y en la programación los algoritmos son mas importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es tan solo un medio para expresar un algoritmo y una computadora es solo un procesador para ejecutarlo.

Tanto el lenguaje de programación como la computadora son los medios para obtener un fin: *"Conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente".*

Dada la importancia del algoritmo en la ciencia de la computación, un aspecto muy importante será su diseño. Las características fundamentales que debe cumplir todo algoritmo son:

- Un algoritmo debe ser **preciso** e indicar el orden de realización de cada paso.
- Un algoritmo debe estar **definido**. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Un algoritmo debe ser **finito**. Si se sigue un algoritmo, se debe terminar en algún momento, o sea debe tener un número finito de pasos.

Todo algoritmo debe describir 3 partes bien diferenciadas:

1)ENTRADA: los ingredientes y utensilios usados

2) PROCESO: elaboración de la receta

3) SALIDA: terminación del plato (por ejemplo, una tortilla de patata).

Un ejemplo de algoritmo para preparar una "tortilla de patata para 2 personas" podría ser:

1 – Coger 4 patatas medianas

2 – Pelar y partir en trozos pequeños las patatas

3 – Salar las patatas ya partidas

4 – Coger una sartén y echarle un poco de aceite

5 – Poner la sartén a calentar

6- Echar las patatas en la sartén a freir

7- MIENTRAS las patatas no estén fritas HACER:

    8- Dejarlas freir

    9- Revolver de vez en cuando

10- Sacar las patatas de la sartén

11 – Coger 4 huevos

12 – Batir los huevos

13– Echar las patatas sobre los huevos batidos

14 – Echar la mezcla de patatas y huevo en la sartén

15 – SI está dorada la tortilla por abajo ENTONCES:

    16 – Darle la vuelta

  SI NO está aún dorada:

    16 – Dejarla un poco más por ese lado

17 – Sacar la tortilla en un plato.

En general, un mismo problema puede ser resuelto por más de un algoritmo.

Un algoritmo se puede representar de varias formas:

GRÁFICA (ordinograma, organigrama, diagrama de NASSI-SCHNEIDERMAN), **ESQUEMÁTICA** (tablas de decisión) **y CON LENGUAJE** (pseudocódigo).

1.4.- Aplicación informática.

Hoy en día la mayoría de los programas que utilizamos o elaboramos están formando parte de un conjunto de programas interrelacionados formando lo que se llama una aplicación informática.

Normalmente se trabaja con la aplicación completa aunque también se puede hacer independientemente con cada uno de los programas que la forman.

1.4.1.- Ciclo de vida de una aplicación informática.

El proceso que se sigue desde el planteamiento de un problema hasta que se tiene una solución instalada en el ordenador se denomina **ciclo de vida** de un sistema informático o **desarrollo** de una aplicación informática.

La forma en que se denomina cada etapa varía según el autor y la metodología seleccionada. La metodología que usaremos nosotros distingue las siguientes fases:

- **Fase de análisis de requerimientos:** En esta fase se comienza por identificar las necesidades del usuario y las especificaciones que debe cumplir la aplicación.

  Es decir, identificamos QUÉ HACE el programa.

  Como resultado de este análisis obtenemos lo que se llama las especificaciones del problema.

  Los ERRORES cometidos en esta fase son los peores de remediar.

  A partir de las especificaciones del problema se pasa a la fase de diseño.

- **Fase de diseño:** En esta fase se determina CÓMO HACE el programa la tarea solicitada en el análisis, es decir, se diseña el algoritmo.

  Para ello, la técnica más usada es DIVIDE Y VENCERÁS. Es decir, se divide el problema entero en subproblemas más pequeños y se buscan algoritmos para cada subproblema. Así, cualquier programa bien diseñado consta de un PROGRAMA PRINCIPAL que llama a SUBPROGRAMAS más pequeños que a su vez pueden llamar a otros subprogramas.

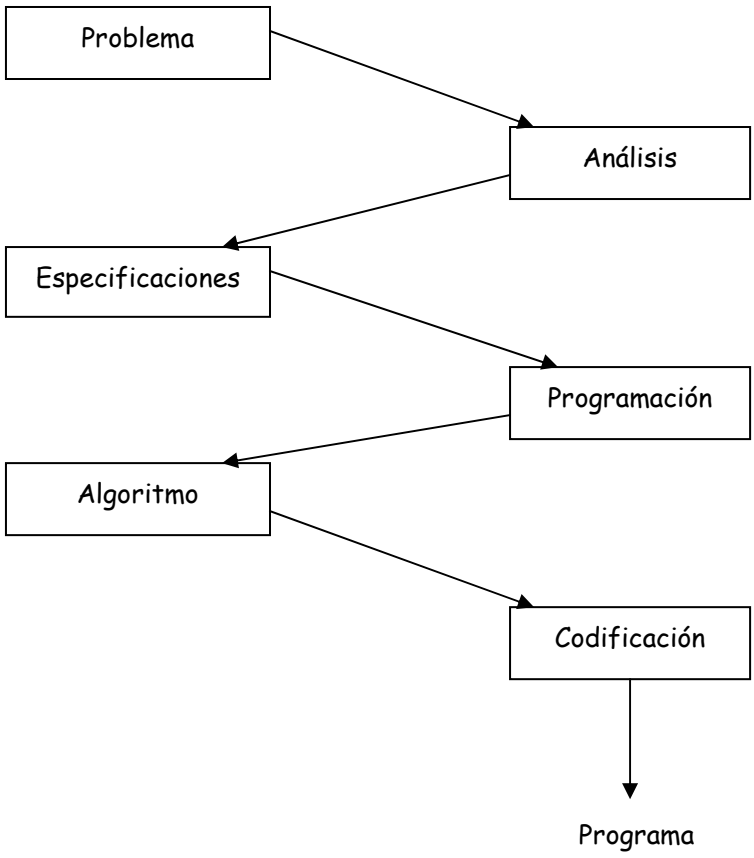
- **Fase de programación:** En esta fase se traduce el algoritmo a un lenguaje de programación.

- **Fase de Prueba del programa:** En esta fase se comprueba si el programa está correctamente instalado en la máquina y si funciona correctamente. Deben tenerse en cuenta todas las posibles combinaciones de datos de entrada para que esté bien probado.

- **Fase de implantación de la aplicación:** Se implanta la aplicación en el sistema final.

- **Fase de mantenimiento**

  Para completar el ciclo de vida hay que realizar las correcciones necesarias para subsanar errores y deficiencias del producto desarrollado, o modificaciones posteriores a su implantación.



.

1.5.- Características de los programas.

Un determinado problema puede tener uno o mas algoritmos que nos permitan llegar a una determinada solución. La elección del algoritmo mas adecuado se debe basar en

una serie de requisitos de calidad que adquieren gran importancia a la hora de evaluar el coste de su diseño y mantenimiento.

Las características generales que debe reunir un programa son las siguientes:

- **Legibilidad:** tendrá que estar escrito de forma que sea fácil su lectura y comprensión.
- **Portabilidad:** su diseño debe permitir la codificación en distintos lenguajes de programación.
- **Modularidad:** para poder adaptarlos a una nueva situación.
- **Eficiencia:** aprovechando al máximo los recursos del ordenador.
- **Estructuración:** debe cumplir las reglas de la "programación estructurada" para facilitar la verificación y depuración del programa.

1.6.- Lenguajes de programación.

Un lenguaje de programación es una notación para escribir programas, a través de los cuales podemos comunicarnos con el hardware del ordenador y dar así las ordenes adecuadas para la realización de un determinado proceso.

Los distintos niveles de programación existentes nos permiten acceder al hardware de tal forma que, según utilicemos un nivel u otro, así tendremos que utilizar un determinado lenguaje. Una clasificación de los lenguajes desde el punto de vista de la programación de aplicaciones es:

- Lenguaje máquina.
- Lenguaje de bajo nivel (ensamblador).
- Lenguajes de alto nivel.

1.6.1.- Lenguajes máquina.

Son aquellos que están escritos en lenguajes directamente inteligibles por la máquina (ordenador), ya que sus instrucciones son cadenas binarias (0, 1) que especifican una operación y las posiciones (direcciones) de memoria implicadas en la operación se denominan *"instrucciones máquina"* o *"código máquina".*

El código máquina es el conocido código máquina.

Las instrucciones del lenguaje máquina dependen del hardware de la computadora y, por tanto, se diferenciara de un ordenador a otro.

- **Ventajas:** Posibilidad de cargar (transferir un programa a la memoria) sin necesidad de traducción posterior, lo que supone una velocidad de ejecución superior a cualquier otro lenguaje de programación.
- **Inconvenientes:**
    - Dificultad y lentitud en la codificación.
    - Poca fiabilidad.
    - Dificultad grande a la hora de verificar y poner a punto los programas.
    - Los programas solo son ejecutables en el mismo procesador.

Los inconvenientes superan ampliamente a las ventajas, lo que hace prácticamente no recomendables a los lenguajes máquina.

1.6.2.- Lenguajes de bajo nivel.

Estos lenguajes son mas fáciles de utilizar que los lenguajes máquina, pero al igual que ellos dependen del ordenador, por lo que no hay un único lenguaje ensamblador.

El lenguaje de bajo nivel por excelencia es el lenguaje ensamblador, (assembler language).

Las instrucciones en lenguaje ensamblador son instrucciones conocidas como nemotécnicos (mnemonics).

Por ejemplo, nemotécnicos típicos de operaciones aritméticas son:

- En Ingles ADD, SUB, DIV
- En Español SUM, RES, DIV

Una instrucción típica de suma seria:

ADD M, N, P

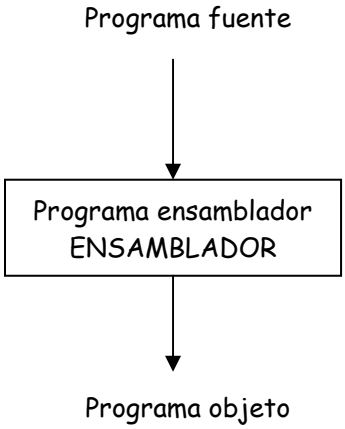
Esta instrucción podría significar: "Sumar el número contenido en la posición de memoria M al número almacenado en la posición de memoria N y situar el resultado en la posición de memoria P".

Evidentemente es mucho mas sencillo recordar la instrucción anterior que su equivalente con código máquina:

0110 1001 1010 1011

Un programa escrito en lenguaje ensamblador no puede ser ejecutado directamente por la computadora, sino que requiere una fase de traducción al lenguaje máquina.

El programa original escrito en lenguaje ensamblador, se denomina programa fuente y el programa traducido en lenguaje máquina se conoce como programa objeto, y es directamente entendible por el ordenador.



- **Ventajas:** Las ventajas de estos lenguajes frente a los lenguajes máquina son:
  - Mayor facilidad de codificación
  - Mayor velocidad de cálculo.
- **Inconvenientes:**
  - Dependencia total de la máquina, lo que impide la transportabilidad de los programas (posibilidad de ejecutar un programa en diferentes máquinas).
  - La formación de los programadores es mas compleja ya que exige no solo técnicas de programación, sino también el conocimiento del interior de la máquina.

1.6.3.- Los lenguajes de alto nivel.

Los lenguajes de alto nivel son los más utilizados por los programadores. Están diseñados para que las personas escriban y entiendan los programas los programas de un modo mucho más fácil que los lenguajes máquina y ensambladores.

Otra razón es que un programa escrito en un lenguaje de alto nivel es independiente de la máquina, es decir que las instrucciones del programa de la computadora no dependen del diseño del hardware o de una computadora en particular. En consecuencia, los programas escritos en lenguajes de alto nivel son transportables, lo que significa que tienen la posibilidad de poder se ejecutados con poca o ninguna modificación en diferentes tipos de computadoras.

- **Ventajas:**

- El tiempo de formación de los programadores es relativamente corto comparado con otros lenguajes.
- La escritura de programas se basa en reglas sintácticas similares a los lenguajes humanos.
- Las modificaciones y puestas a punto de los programas son más fáciles.
- Reducción del coste de los programas.
- Transportabilidad.

➡ **Inconvenientes:**

- Incremento del tiempo de puesta a punto al necesitarse diferentes traducciones del programa fuente para conseguir el programa definitivo.
- No se aprovechan los recursos internos de la máquina que se explotan mucho mejor en lenguajes máquina y ensambladores.
- Aumento de la ocupación de memoria.
- El tiempo de ejecución de los programas es mucho mayor.

Al igual que sucede con los lenguajes ensambladores, los lenguajes fuente tienen que ser traducidos por programas traductores, llamados en este caso compiladores e interpretes.

Los lenguajes de programación de alto nivel existentes en la actualidad son muy numerosos, entre ellos están BASIC, COBOL, C, C++, PASCAL, etc.

1.7.- Traductores del lenguaje.

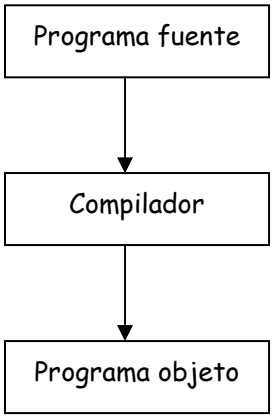
Los traductores de lenguaje son programas que traducen a su vez los programas fuente escritos en lenguajes de alto nivel a código máquina. Los traductores se dividen en:

- Compiladores.
- Intérpretes.

1.7.1.- Compiladores.

Un compilador es un programa que traduce los programas fuente escritos en lenguajes de alto nivel a lenguaje máquina.

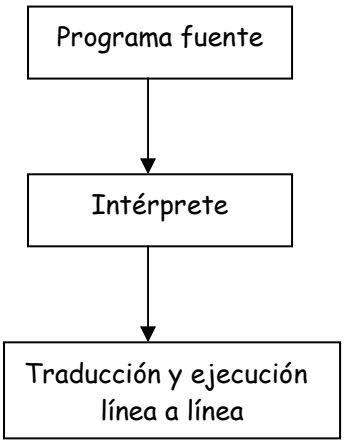
El programa escrito en lenguajes de alto nivel se llaman programa fuente y el programa traducido programa objeto o código objeto. El compilador traduce sentencia a sentencia el programa fuente.



1.7.2.- Intérpretes.

Un intérprete es un traductor que toma un programa fuente, lo traduce y a continuación lo ejecuta.

Un lenguaje que soporte un traductor de tipo intérprete se denomina lenguaje interpretado. BASIC es el modelo por excelencia de lenguaje interpretado.



1.8.- La compilación y sus fases.

La compilación es el proceso de traducción de programas fuente a programas objeto.

El programa objeto obtenido de la compilación no ha sido traducido normalmente a código máquina sino a ensamblador.

Para conseguir el programa máquina real se debe utilizar un programa llamado montador o enlazador (linker). El proceso de montaje conduce a un programa en lenguaje máquina directamente ejecutable.

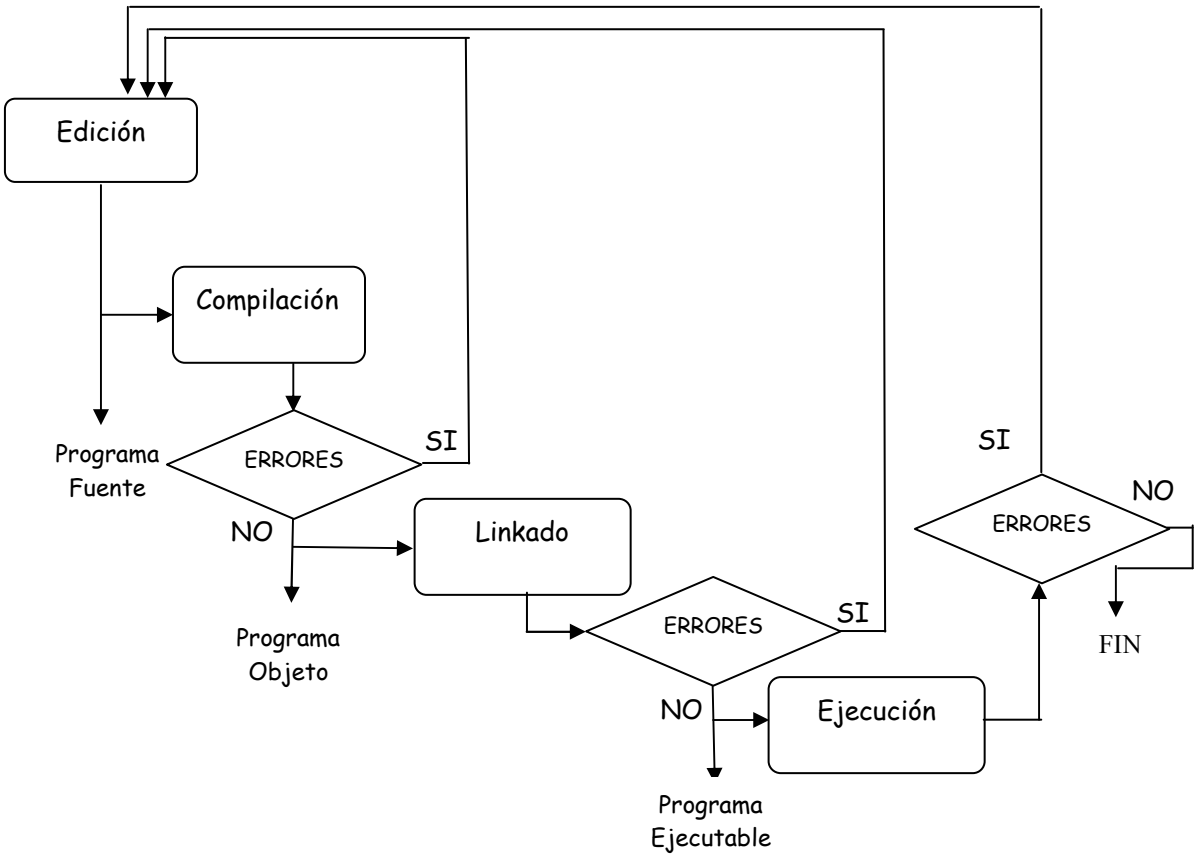
El proceso de ejecución de un programa en un lenguaje de alto nivel sigue por tanto las siguientes fases:

- **Edición:** Consiste en la escritura del programa (empleando un lenguaje de programación previamente seleccionado) y su posterior grabación sobre un soporte de almacenamiento permanente. La edición del programa debe realizarse mediante la utilización de un editor, que puede formar parte o no del compilador utilizado. En esta fase se obtiene el denominado programa fuente.

- **Compilación:** En esta fase se traduce el programa fuente a su equivalente código máquina, obteniendo en caso de que no se produzca ningún error el denominado programa objeto. En caso de producirse errores, el compilador los mostrará utilizando los mensajes correspondientes, que nos permitirán corregir el programa fuente y proceder de nuevo a su compilación.

- **Linkado:** Esta fase también recibe el nombre de montaje y consiste en unir o enlazar el programa objeto obtenido en la fase de compilación con determinadas rutinas internas del lenguaje y, si el método de programación es modular, se enlazan los distintos módulos para obtener así el programa ejecutable.

- **Ejecución:** Esta fase consiste en la llamada del programa ejecutable a través del sistema operativo. Inicialmente se debe comprobar el buen funcionamiento del programa mediante el uso de unos juegos de pruebas que especifican los resultados que se desean obtener en función de unos determinados datos de entrada.



Los principales errores en la ejecución de un programa son:

- Datos de entrada incorrectos que producen una parada del sistema (por ejemplo, introducir un dividendo con valor cero en una operación de división).
- Bucles mal definidos que producen un funcionamiento continuo del programa (por ejemplo, un bucle sin fin o bucle infinito).
- Datos de salida incorrectos, producidos por un mal desarrollo del programa o ambigüedad en las especificaciones del usuario.

1.9.- Tipos de ERRORES que se pueden producir.

Cuando se desarrolla una aplicación, se pueden producir errores, que pueden ser de varios tipos:

A) ERRORES DE COMPILACIÓN: Son errores de sintaxis del lenguaje elegido.

B) ERRORES DE EJECUCIÓN: Se producen por instrucciones que la computadora puede comprender, pero no ejecutar. Por ejemplo una división por cero. En estos casos, la ejecución del programa se detiene.

C) ERRORES LÓGICOS: Se producen en la lógica del programa y suelen ser debidos a fallos en el diseño del algoritmo. Son los más difíciles de detectar, ya que el programa suele funcionar sin dar errores, pero obtiene resultados no deseados. En este caso se debe volver a la fase de diseño, modificarlo y cambiar el programa fuente y volver a compilar y ejecutar.

Los errores de COMPILACIÓN y EJECUCIÓN pueden ser corregidos y depurados por programas DEBUGGERS o COMPILADORES.

1.10- Documentación de los programas.

Todas las aplicaciones debe ser documentadas, así como todos los programas que las constituyen.

NO HAY QUE ESPERAR A TENER FINALIZADO EL PROGRAMA PARA EMPEZAR A DOCUMENTAR, sino que hay que realizar la documentación **EN CADA UNA DE LAS FASES DEL CICLO DE VIDA.**

Normalmente la documentación incluye:

-GUÍA TÉCNICA

-GUÍA DE USO O UTILIZACIÓN

-GUÍA DE INSTALACIÓN

1.10.1.-GUÍA TÉCNICA

QUE ES:

Refleja el ***diseño de la aplicación, la codificación de los programas y las pruebas*** realizadas para su correcto uso.

A QUIEN VA DESTINADO:

Personal técnico informático (analistas y programadores)

QUÉ DOCUMENTOS INCLUYE:

-cuaderno de carga

(refleja el diseño y análisis de la aplicación, destinado a programadores. Incluye el diseño de datos, entradas y salidas, pantallas,…)

-programa fuente (código de los programas con autodocumentación, destinado a programadores y analistas)

-juego de pruebas (incluye las pruebas realizadas, destinado a programadores y analistas)

1.10.2.-GUÍA DE USO O UTILIZACIÓN

QUE ES:

Es el manual de usuario. Contiene toda la información necesaria para que el usuario pueda funcionar adecuadamente con la aplicación.

A QUIÉN VA DESTINADO:

A los usuarios de la aplicación.

1.10.3.-GUÍA DE INSTALACIÓN

QUE ES:

Es el manual de explotación. Contiene toda la información necesaria para la puesta en marcha del sistema diseñado y normas de explotación.